EJERCICIOS PL/SQL : BLOQUES

Dada la base de datos empresa que se encuentra en el fichero creaBD_empresa.sql y que tiene el siguiente esquema relacional:

DEPART (dept_no, dnombre, loc)
EMPLE (emp no, apellido, oficio, dir, fecha_alt, salario, comision,)
Clave ajena: dir → EMPLE.emp_no
Clave ajena: dept_no → DEPART.dept_no

1. **Crea un bloque PL/SQL que cuente el número de empleados, deposite su resultado en una variable y lo muestre por pantalla.**

```
DECLARE
  v_total_empleados NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_total_empleados FROM emple;
  DBMS_OUTPUT.PUT_LINE('Número total de empleados: ' || v_total_empleados);
END;
/
```

2. **Diseña un bloque que nos muestre el departamento al que pertenece el empleado SALA.**

```
DECLARE
  v_departamento VARCHAR2(40);
BEGIN
  SELECT dnombre INTO v_departamento
  FROM emple e
JOIN depart d ON e.dept_no = d.dept_no
  WHERE e.apellido = 'SALA';

  DBMS_OUTPUT.PUT_LINE('El empleado SALA pertenece al departamento: ' ||
v_departamento);
END;
/
```

3. **Diseñar un bloque de programa que muestre el emp_no que le correspondería a un nuevo empleado, sabiendo que sería el siguiente valor al mayor de todos los emp_no de la tabla EMPLE.**

```
DECLARE
  v_nuevo_emp_no NUMBER;
BEGIN
  SELECT NVL(MAX(emp_no), 0) + 1 INTO v_nuevo_emp_no FROM emple;
  DBMS_OUTPUT.PUT_LINE('El próximo emp_no disponible es: ' || v_nuevo_emp_no);
END;
```

Solución Ana Ej3:

```
DECLARE
   num_max emple.emp_no%TYPE;
BEGIN
   SELECT MAX(emp_no) + 1 INTO num_max FROM emple;
   DBMS_OUTPUT.PUT_LINE('El próximo emp_no disponible es: ' || num_max);
END;
/
```

4. **Bloque que calcule cual es el departamento cuyo salario medio es menor. Visualizar el nombre de departamento y su salario medio.**

```
DECLARE
   v_dept_no NUMBER;
   v_nombre_departamento VARCHAR2(40);
   v_salario_medio NUMBER(7,2);
BEGIN
   SELECT dept_no, dnombre, AVG(salario)
   INTO v_dept_no, v_nombre_departamento, v_salario_medio
   FROM emple JOIN depart USING (dept_no)
   GROUP BY dept_no, dnombre
   ORDER BY AVG(salario)
   FETCH FIRST ROW ONLY;

   DBMS_OUTPUT.PUT_LINE('El departamento con menor salario medio es: ' ||
v_nombre_departamento ||
                        ' con un salario medio de ' || v_salario_medio);
END;
/
```

Solucion Jorge:

```
DECLARE
   v_nombre_departamento VARCHAR2(40);
   v_salario_medio NUMBER(7,2);
BEGIN
   SELECT dnombre, AVG(e.salario)
   INTO v_nombre_departamento, v_salario_medio
   FROM depart d
JOIN emple ON d.depart_no = e.depart_no
   GROUP BY dept_no, dnombre
   ORDER BY AVG(salario)
   FETCH FIRST 1 ROWS ONLY;

   DBMS_OUTPUT.PUT_LINE('Depart. con menor salario medio es: ' || v_nombre_departamento
|| ' con un salario medio de ' || v_salario_medio);
END;
/
```

Solución Profe:

```
DECLARE
   v_nombre_departamento VARCHAR2(40);
   v_salario_medio NUMBER(7,2);
BEGIN
   SELECT dnombre, AVG(e.salario)
   INTO v_nombre_departamento, v_salario_medio
   FROM depart d
JOIN emple e ON d.dept_no = e.dept_no
   GROUP BY dnombre
HAVING AVG(e.salario) = (SELECT MIN(AVG(salario)) FROM emple GROUP BY dept_no);

   DBMS_OUTPUT.PUT_LINE('Depart. con menor salario_m: ' || v_nombre_departamento || '
salario_m: ' || v_salario_medio);
END;
/
```

5. **Diseñar un bloque PL/SQL que permita insertar una fila con la siguiente información:**
    - **Emp_no: siguiente valor al mayor de todos los emp_no.**
    - **Apellido: se introducirá por teclado tu apellido. Ver nota ***
    - **Oficio: ANALISTA y se introducirá por teclado. Ver nota ***
    - **Dept_no y salario: se calcularán a partir del ejercicio anterior.**
    - **Jefe (DIR): emp_no del DIRECTOR de departamento donde se dará de alta.**
    - **Fecha de alta: la fecha actual.**
    - **El resto de los campos se dejan en blanco.**

* Nota: si usáis Oracle live, al insertar por teclado veréis que da error debido a que insertar de esta manera está limitado en este servicio. Como solución, podéis poner la instrucción por teclado entre comentarios e introducir el valor en una variable de modo normal y, cuando tengáis acceso a oracle server, verificar que es correcto.

```
DECLARE
   v_nuevo_emp_no NUMBER;
   v_apellido VARCHAR2(40) := 'TU_APELLIDO'; -- Reemplázalo con tu apellido
   v_oficio VARCHAR2(50) := 'ANALISTA';
   v_dept_no NUMBER;
   v_salario NUMBER(7,2);
   v_jefe NUMBER;
BEGIN
   -- Obtener el próximo emp_no
   SELECT NVL(MAX(emp_no), 0) + 1 INTO v_nuevo_emp_no FROM emple;

   -- Obtener el departamento con menor salario medio
   SELECT dept_no, AVG(salario)
   INTO v_dept_no, v_salario
   FROM emple
```

```
    GROUP BY dept_no
    ORDER BY AVG(salario)
    FETCH FIRST ROW ONLY;

    -- Obtener el emp_no del director del departamento elegido
    SELECT emp_no INTO v_jefe
    FROM emple
    WHERE dept_no = v_dept_no AND oficio = 'DIRECTOR';

    -- Insertar nuevo empleado
    INSERT INTO emple (emp_no, apellido, oficio, dir, fecha_alt, salario, dept_no)
    VALUES (v_nuevo_emp_no, v_apellido, v_oficio, v_jefe, SYSDATE, v_salario, v_dept_no);

    DBMS_OUTPUT.PUT_LINE('Empleado insertado correctamente con emp_no: ' ||
v_nuevo_emp_no);
END;
/
```

Solución Iñigo:

```
DECLARE
    media NUMBER;
    minDept NUMBER;
    apellido VARCHAR(40);
    oficio emple.oficio%TYPE;
BEGIN
    apellido := '$ApellidoTeclado';
    oficio := '$oficio'; -- Por teclado se introduce ANALISTA
    Select e.dept_no, AVG(e.salario) INTO minDept, media
    from emple e
    group by e.dept_no
    having AVG(e.salario) =  (Select MIN(AVG(salario)) from emple group by dept_no);
    INSERT INTO emple VALUES(
      (Select MAX(emp_no + 1) from emple), apellido, oficio, (Select emp_no from emple where
oficio = 'DIRECTOR' AND dept_no = minDept), SYSDATE, media, 0, minDept);
    END;
/
```

6. **Crear una tabla llamada DATOS_EMPLE con la siguiente información:**
    - **Emp_no clave primaria**
    - **Apellido**
    - **Sueldo**
    - **Estrellas: donde por cada 500€ se le asignará una estrella al empleado (cada una será representada por el carácter *).**
    - **Si se elimina un empleado en la tabla EMPLE, sus datos en esta tabla también deben hacerlo.**

```
CREATE TABLE datos_emple (
   emp_no NUMBER(9) PRIMARY KEY,
   apellido VARCHAR2(40),
   sueldo NUMBER(7,2),
   estrellas VARCHAR2(10),
   CONSTRAINT fk_datos_emple FOREIGN KEY (emp_no) REFERENCES emple(emp_no) ON
DELETE CASCADE
);
```

7. **Crear un bloque PL/SQL que una vez introduzca un número de empleado, localice dicho empleado en la tabla EMPLE e inserte en la tabla DATOS_EMPLE la información correspondiente al emp_no, apellido, sueldo.**

```
DECLARE
   v_emp_no NUMBER;
   v_apellido VARCHAR2(40);
   v_sueldo NUMBER(7,2);
   v_estrellas VARCHAR2(50);
BEGIN
   v_emp_no := &Ingrese_Empleado_ID; -- Se debe ingresar manualmente en Oracle Live SQL

   SELECT apellido, salario INTO v_apellido, v_sueldo
   FROM emple
   WHERE emp_no = v_emp_no;

   -- Calcular estrellas
   v_estrellas := LPAD('*', FLOOR(v_sueldo / 500), '*');

   -- Insertar en DATOS_EMPLE
   INSERT INTO datos_emple (emp_no, apellido, sueldo, estrellas)
   VALUES (v_emp_no, v_apellido, v_sueldo, v_estrellas);

END;
/
```

Solución Profe:

```
SET SERVEROUTPUT ON
DECLARE
        numemple        emple.emp_no%TYPE;
        apellido        emple.apellido%TYPE;
        sueldo          emple.salario%TYPE;
        estrellas       datos_emple.estrellas%TYPE DEFAULT ‘’; -- Por defecto estrellas tiene ‘’.
        n               NUMBER
BEGIN
        numemple := &num; -- Símbolo & es para introducir por teclado
        SELECT apellido, salario + comision INTO apellido, sueldo
        FROM emple
        WHERE emple_no = numemple;
        n := FLOOR (sueldo/500); -- Floor es para redondear hacia abajo (Truncar)
        FOR i IN 1..n LOOP
                estrellas := CONCAT(estrellas, ‘*’);
        END LOOP;
        INSERT INTO datos_emple VALUES (numemple, apellido, sueldo, estrellas);
END;
/
```

PROCEDIMIENTOS Y FUNCIONES

1. Procedimientos:

a) Escribir un procedimiento que reciba por parámetro un número y permita calcular el doble y la mitad de un número.

```
CREATE OR REPLACE PROCEDURE calcular_doble_mitad (p_num NUMBER)
IS
  v_doble NUMBER;
  v_mitad NUMBER;
BEGIN
  v_doble := p_num * 2;
  v_mitad := p_num / 2;
  DBMS_OUTPUT.PUT_LINE('El doble de ' || p_num || ' es ' || v_doble);
  DBMS_OUTPUT.PUT_LINE('La mitad de ' || p_num || ' es ' || v_mitad);
END;
/
```

b) Prueba a realizar el mismo ejemplo realizando un procedimiento para calcular el doble, una función para calcular la mitad y desde un bloque llame a ambos.

```
CREATE OR REPLACE PROCEDURE doble(p_num NUMBER)
IS
  v_doble NUMBER;
BEGIN
  v_doble := p_num * 2;
  DBMS_OUTPUT.PUT_LINE('El doble de ' || p_num || ' es ' || v_doble);
END;
/

CREATE OR REPLACE FUNCTION fun_mitad (p_num NUMBER)
RETURN NUMBER IS
BEGIN
  RETURN p_num / 2;
END;
/
DECLARE
  v_num   NUMBER := 10;  -- valor de ejemplo
  v_mitad NUMBER;
BEGIN
  proc_doble(v_num);
  v_mitad := fun_mitad(v_num);
  DBMS_OUTPUT.PUT_LINE('La mitad de ' || v_num || ' es ' || v_mitad);
END; /
```

2. Diseñar una función recursiva que permita calcular el término enésimo de la sucesión de Fibonacci, es decir, si el parámetro recibido de la función es 0 devolverá 0, si es 1 devolverá 1 y cualquier otro término se calcula como la suma de los dos anteriores.

Fibonacci (0) = 0
Fibonacci (1) = 1
Fibonacci (n) = Fibonacci (n-1) + Fibonacci (n-2)

```
CREATE OR REPLACE FUNCTION fibonacci(n NUMBER)
RETURN NUMBER
IS
BEGIN
  IF n = 0 THEN
    RETURN 0;
  ELSIF n = 1 THEN
    RETURN 1;
  ELSE
    RETURN fibonacci(n - 1) + fibonacci(n - 2);
  END IF;
END;
/
```

3. Escribir un bloque que permita escribir los 'n' primeros términos de la sucesión de Fibonacci quedando de la siguiente forma: 0, 1, 1, 2, 3, 5, 8, 13, 21,...

¿Qué habría que hacer para que quedara del revés?: 55 34 21 13853210

```
DECLARE
  n             NUMBER := 10;  -- Número de términos a mostrar (se puede modificar)
  i             NUMBER;
  v_fib         NUMBER;
  v_secuencia   VARCHAR2(4000) := '';
BEGIN
  -- Imprimir la secuencia en orden natural: 0, 1, 1, 2, 3, 5, 8, 13, 21,...
  FOR i IN 0..n-1 LOOP
    v_fib := fibonacci(i);
    IF i < n-1 THEN
      v_secuencia := v_secuencia || v_fib || ', ';
    ELSE
      v_secuencia := v_secuencia || v_fib;
    END IF;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Secuencia Fibonacci: ' || v_secuencia);

  -- Para imprimir la secuencia en orden inverso (del último al primero)
  v_secuencia := '';
  FOR i IN REVERSE 0..n-1 LOOP
    v_fib := fibonacci(i);
    IF i > 0 THEN
      v_secuencia := v_secuencia || v_fib || ', ';
    ELSE
      v_secuencia := v_secuencia || v_fib;
    END IF;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Secuencia Fibonacci (inversa): ' || v_secuencia);
END;
/
```

CURSORES

4. Realiza un procedimiento verEmpleados que muestre los apellidos y la fecha de alta de los empleados de un determinado departamento pasado por parámetro. En el caso de tener un salario inferior a 1200 € se deberá mostrar el mensaje "salario bajo", si está entre 1200 y 3000€ mostrará "salario medio" y si es mayor de 3000€ mostrará "salario alto" y además, en este último caso se visualizará también su oficio.
Por último, deberá de mostrar un mensaje con el número total de empleados del departamento.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE verEmpleados (departamento emple.dept_no%TYPE)
IS
   CURSOR c_empleados IS
      SELECT apellido, fecha_alt, salario, oficio
      FROM emple
      WHERE dept_no = departamento;

   v_apellido emple.apellido%TYPE;
   v_fecha_alt emple.fecha_alt%TYPE;
   v_salario emple.salario%TYPE;
   v_oficio emple.oficio%TYPE;
   v_num_empleados := 0;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Empleados del departamento ' || departamento || ':');

   FOR emp IN c_empleados LOOP
      v_num_empleados := v_num_empleados + 1;
      v_apellido := emp.apellido;
      v_fecha_alt := emp.fecha_alt;
      v_salario := emp.salario;
      v_oficio := emp.oficio;

      IF v_salario < 1200 THEN
         DBMS_OUTPUT.PUT_LINE(v_apellido || ' - ' || v_fecha_alt || ' - Salario bajo');
      ELSIF v_salario BETWEEN 1200 AND 3000 THEN
         DBMS_OUTPUT.PUT_LINE(v_apellido || ' - ' || v_fecha_alt || ' - Salario medio');
      ELSE
         DBMS_OUTPUT.PUT_LINE(v_apellido || ' - ' || v_fecha_alt || ' - Salario alto - Oficio: ' ||
v_oficio);
      END IF;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Total empleados en el departamento: ' || v_num_empleados);
END verEmpleados;
/
```

Solución Clase (Yoel):

```
CREATE OR REPLACE PROCEDURE verEmpleados (dept_num IN NUMBER) IS
   CURSOR c_empleados IS
      SELECT apellido, fecha_alt, salario, oficio
      FROM empleado;
      WHERE dept_no = dept_num;
      ORDER BY APELLIDO;

   v_num_empleados := 0;

BEGIN
   FOR emp IN c_empleados LOOP
      v_num.empleados := v_num_empleados + 1;

      IF emp.salario < 1200 THEN
         DBMS_OUTPUT.PUT_LINE(v_apellido || ' - ' || v_fecha_alt || ' - Salario bajo');
      ELSIF emp.salario BETWEEN 1200 AND 3000 THEN
         DBMS_OUTPUT.PUT_LINE(v_apellido || ' - ' || v_fecha_alt || ' - Salario medio');
      ELSE
         DBMS_OUTPUT.PUT_LINE(v_apellido || ' - ' || v_fecha_alt || ' - Salario alto - Oficio: ' ||
v_oficio);
      END IF;
   END LOOP;

   DBMS_OUTPUT.PUT_LINE('Número total empleados: ' || v_num_empleados);
END;
/
```

5. Realiza un procedimiento verDptoEmpleados que visualice todos los departamentos de la empresa, se deberá de mostrar su nombre y localización, así como los empleados que tiene cada departamento mostrando los mismos datos que en el ejercicio anterior.
Se desea saber el número de empleados que tiene cada departamento, el número total de empleados de la empresa y el número total de departamentos de la empresa.

```
CREATE OR REPLACE PROCEDURE verDptoEmpleados IS
  CURSOR c_depart IS
    SELECT dept_no, dnombre, loc FROM depart;
  CURSOR c_empleados(p_dept_no emple.dept_no%TYPE) IS
    SELECT apellido, fecha_alt, salario, oficio
    FROM emple
    WHERE dept_no = p_dept_no;

  v_total_empleados NUMBER := 0;
  v_total_dptos NUMBER := 0;
BEGIN
  FOR d IN c_depart LOOP
    v_total_dptos := v_total_dptos + 1;
    DBMS_OUTPUT.PUT_LINE('Departamento: ' || d.dnombre || ' - ' || d.loc);

    DECLARE
      v_contador NUMBER := 0;
    BEGIN
      FOR emp IN c_empleados(d.dept_no) LOOP
        v_contador := v_contador + 1;
        v_total_empleados := v_total_empleados + 1;

        IF emp.salario < 1200 THEN
          DBMS_OUTPUT.PUT_LINE(emp.apellido || ' - ' || emp.fecha_alt || ' - Salario bajo');
        ELSIF emp.salario BETWEEN 1200 AND 3000 THEN
          DBMS_OUTPUT.PUT_LINE(emp.apellido || ' - ' || emp.fecha_alt || ' - Salario medio');
        ELSE
          DBMS_OUTPUT.PUT_LINE(emp.apellido || ' - ' || emp.fecha_alt || ' - Salario alto -
Oficio: ' || emp.oficio);
        END IF;
      END LOOP;
      DBMS_OUTPUT.PUT_LINE('Total empleados en ' || d.dnombre || ': ' || v_contador);
    END;
  END LOOP;

  DBMS_OUTPUT.PUT_LINE('Total departamentos: ' || v_total_dptos);
  DBMS_OUTPUT.PUT_LINE('Total empleados en la empresa: ' || v_total_empleados);
END verDptoEmpleados;
/
```

6. Realiza un procedimiento utilizando cursores que visualice el apellido y el salario de los cinco empleados con el salario más alto.

```
CREATE OR REPLACE PROCEDURE top5Salarios IS
  CURSOR emp5 IS
    SELECT apellido, salario
    FROM emple
    ORDER BY salario DESC
    FETCH FIRST 5 ROWS ONLY;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Top 5 empleados con mayor salario:');
  FOR emp IN emp5 LOOP
    DBMS_OUTPUT.PUT_LINE(emp.apellido || ' - ' || emp.salario);
  END LOOP;
END top5Salarios;
/
```

7. Codificar un programa que visualice los dos empleados que ganan menos de cada oficio.

```
CREATE OR REPLACE PROCEDURE bottom2Salarios IS
  CURSOR c_oficios IS
    SELECT DISTINCT oficio FROM emple;

  CURSOR c_empleados(p_oficio emple.oficio%TYPE) IS
    SELECT apellido, salario
    FROM emple
    WHERE oficio = p_oficio
    ORDER BY salario ASC
    FETCH FIRST 2 ROWS ONLY;
BEGIN
  FOR o IN c_oficios LOOP
    DBMS_OUTPUT.PUT_LINE('Oficio: ' || o.oficio);
    FOR emp IN c_empleados(o.oficio) LOOP
      DBMS_OUTPUT.PUT_LINE(emp.apellido || ' - ' || emp.salario);
    END LOOP;
  END LOOP;
END;
/
```

8. Realiza un sólo procedimiento que muestre todos los departamentos y en cada departamento los empleados que tiene ordenados por apellido.

- **Se mostrará para cada empleado el apellido y sueldo.**
- **De cada departamento: el nombre, número de empleados y salario medio.**
- **Al final del listado se mostrará el número total de empleados y la suma de todos los salarios.**

```
CREATE OR REPLACE PROCEDURE listar_departamentos_empleados IS
  CURSOR dept_cursor IS
    SELECT d.dept_no, d.dnombre
    FROM depart d;

  CURSOR emp_cursor(p_dept_no depart.dept.no%TYPE) IS
    SELECT apellido, salario
    FROM emple
    WHERE dept_no = p_dept_no
    ORDER BY apellido;

  v_total_empleados NUMBER := 0;
  v_total_salarios NUMBER := 0;

BEGIN
  FOR dept_rec IN dept_cursor LOOP
    DECLARE
      v_num_empleados NUMBER := 0;
      v_suma_salarios NUMBER := 0;
      v_salario_medio NUMBER;
    BEGIN
      DBMS_OUTPUT.PUT_LINE('Departamento: ' || dept_rec.dnombre);

      FOR emp_rec IN emp_cursor(dept_rec.dept_no) LOOP
        DBMS_OUTPUT.PUT_LINE(' - ' || emp_rec.apellido || ' | Salario: ' || emp_rec.salario);
        v_num_empleados := v_num_empleados + 1;
        v_suma_salarios := v_suma_salarios + emp_rec.salario;
      END LOOP;

      IF v_num_empleados > 0 THEN
        v_salario_medio := v_suma_salarios / v_num_empleados;
      ELSE
        v_salario_medio := 0;
      END IF;

      DBMS_OUTPUT.PUT_LINE('Número de empleados: ' || v_num_empleados);
      DBMS_OUTPUT.PUT_LINE('Salario medio: ' || v_salario_medio);
```

```
        v_total_empleados := v_total_empleados + v_num_empleados;
        v_total_salarios := v_total_salarios + v_suma_salarios;
      END;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Total empleados en la empresa: ' || v_total_empleados);
  DBMS_OUTPUT.PUT_LINE('Suma de todos los salarios: ' || v_total_salarios);
END;
/
```

Solución Profe:

```
CREATE OR REPLACE PROCEDURE listar_departamentos_empleados IS
  CURSOR dept_cursor IS
    SELECT d.dept_no, d.dnombre
    FROM depart d;

  CURSOR emp_cursor(p_dept_no depart.dept.no%TYPE) IS
    SELECT apellido, salario
    FROM emple
    WHERE dept_no = p_dept_no
    ORDER BY apellido;

        total_empleados NUMBER := 0;
        suma_salarios NUMBER := 0;
        salario_medio NUMBER;

BEGIN
  FOR dept_rec IN dept_cursor LOOP
    SELECT AVG(salario) INTO salario_medio
    FROM emple
    WHERE dept_no = dept_rec.dept_no

    SELECT COUNT(*) INTO total_empleados
    FROM emple
    WHERE dept_no = dept_rec.dept_no
    DBMS_OUTPUT-PUT_LINE (dept_rec.nombre || total_empleados || salario_medio || '€')

          FOR fila_emp IN c_empleados (fila_dept.dept_no) LOOP
            DBMS_OUTPUT.PUT_LINE(fila_emp.apellido || fila_emp.salario);
          END LOOP;
   END LOOP;
END;
/
```

9. Escribir un procedimiento que suba el sueldo a todos los empleados que ganen menos que el salario medio de su oficio. La subida será del 50% de la diferencia entre el salario del empleado y la media de su oficio.

```
CREATE OR REPLACE PROCEDURE subir_sueldo IS
BEGIN
   FOR emp IN (SELECT emp_no, salario, oficio FROM emple) LOOP
      DECLARE
         v_salario_medio NUMBER;
         v_nuevo_sueldo NUMBER;
      BEGIN
         SELECT AVG(salario) INTO v_salario_medio FROM emple WHERE oficio = emp.oficio;
         IF emp.salario < v_salario_medio THEN
            v_nuevo_sueldo := emp.salario + (0.5 * (v_salario_medio - emp.salario));
            UPDATE emple SET salario = v_nuevo_sueldo WHERE emp_no = emp.emp_no;
         END IF;
      END;
   END LOOP;
   COMMIT;
END;
/
```

Solución Iñigo:

```
CREATE OR REPLACE PROCEDURE subir_sueldo IS
CURSOR emp IS SELECT emp_no, salario, oficio FROM emple;
salarioMedio NUMBER := 0;
BEGIN
   FOR i IN emp LOOP
         SELECT AVG(emple.salario) INTO salarioMedio
          FROM emple
          WHERE i.oficio = emple.oficio;
         IF i.salario < emple.salario THEN
            UPDATE emple SET salario = salario + (salarioMedio - salario / 2)
            WHERE emple.emp_no = i.emp_no;
         END IF;
   END LOOP;
END;
/
```

10. Diseñar un procedimiento que simule un listado de liquidación de todos los empleados según la siguientes especificaciones:

****

Liquidación del empleado:..... Departamento:.....
Oficio :.....
Salario :.....
Trienios :.....
Comp. Responsabilidad :.....
Comisión
Total

****

Un trienio son 3 años completos desde la fecha de alta hasta la actual, y supone 30€ por trienio.
El complemento de responsabilidad será de 60€ por cada empleado que se encuentre a cargo del empleado.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE liquidacion_empleados IS
  CURSOR emp_cursor IS
    SELECT emp_no, apellido, oficio, salario, dept_no, fecha_alt, dir FROM emple;

  v_trienios NUMBER;
  v_comp_responsabilidad NUMBER;
  v_total NUMBER;
  v_dep_nombre VARCHAR2(40);
BEGIN
  FOR emp_rec IN emp_cursor LOOP
    SELECT dnombre INTO v_dep_nombre FROM depart WHERE dept_no = emp_rec.dept_no;

    v_trienios := FLOOR(MONTHS_BETWEEN(SYSDATE, emp_rec.fecha_alt) / 36) * 30;
    SELECT COUNT(*) * 60 INTO v_comp_responsabilidad FROM emple WHERE dir =
emp_rec.emp_no;

    v_total := emp_rec.salario + v_trienios + v_comp_responsabilidad +
NVL(emp_rec.comision, 0);

    DBMS_OUTPUT.PUT_LINE('Liquidación del empleado: ' || emp_rec.apellido);
    DBMS_OUTPUT.PUT_LINE('Departamento: ' || v_dep_nombre);
    DBMS_OUTPUT.PUT_LINE('Oficio: ' || emp_rec.oficio);
    DBMS_OUTPUT.PUT_LINE('Salario: ' || emp_rec.salario);
    DBMS_OUTPUT.PUT_LINE('Trienios: ' || v_trienios);
    DBMS_OUTPUT.PUT_LINE('Comp. Responsabilidad: ' || v_comp_responsabilidad);
    DBMS_OUTPUT.PUT_LINE('Comisión: ' || NVL(emp_rec.comision, 0));
    DBMS_OUTPUT.PUT_LINE('Total: ' || v_total);
```

```
    DBMS_OUTPUT.PUT_LINE('****************************************');
  END LOOP;
END;
/
```

11. Modifica el ejercicio anterior para que la liquidación se realice sobre un empleado que es pasado por parámetro.

- **Debe de gestionar las posibles excepciones que puedan surgir, como por ejemplo, que el empleado no exista.**

```
CREATE OR REPLACE PROCEDURE liquidacion_empleado(p_emp_no INT) IS
  v_emp emple%ROWTYPE;
  v_dep_nombre VARCHAR2(40);
  v_trienios NUMBER;
  v_comp_responsabilidad NUMBER;
  v_total NUMBER;
BEGIN
  SELECT * INTO v_emp FROM emple WHERE emp_no = p_emp_no;

  SELECT dnombre INTO v_dep_nombre FROM depart WHERE dept_no = v_emp.dept_no;

  v_trienios := FLOOR(MONTHS_BETWEEN(SYSDATE, v_emp.fecha_alt) / 36) * 30;
  SELECT COUNT(*) * 60 INTO v_comp_responsabilidad FROM emple WHERE dir =
v_emp.emp_no;

  v_total := v_emp.salario + v_trienios + v_comp_responsabilidad + NVL(v_emp.comision, 0);

  DBMS_OUTPUT.PUT_LINE('Liquidación del empleado: ' || v_emp.apellido);
  DBMS_OUTPUT.PUT_LINE('Departamento: ' || v_dep_nombre);
  DBMS_OUTPUT.PUT_LINE('Oficio: ' || v_emp.oficio);
  DBMS_OUTPUT.PUT_LINE('Salario: ' || v_emp.salario);
  DBMS_OUTPUT.PUT_LINE('Trienios: ' || v_trienios);
  DBMS_OUTPUT.PUT_LINE('Comp. Responsabilidad: ' || v_comp_responsabilidad);
  DBMS_OUTPUT.PUT_LINE('Comisión: ' || NVL(v_emp.comision, 0));
  DBMS_OUTPUT.PUT_LINE('Total: ' || v_total);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Error: El empleado con número ' || p_emp_no || ' no existe.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error desconocido: ' || SQLERRM);
END;
/
```

12. Modificar el ejercicio anterior para que, en lugar de visualizar los datos en pantalla, guarde los datos en una nueva tabla T_LIQUIDACION.
Esta nueva tabla se ha creado previamente con las columnas, apellido, departamento, oficio, salario, trienios, comp_responsabilidad, comision y total.
Se debe de controlar las posibles excepciones que puedan surgir, como por ejemplo,

- **Insertar un registro con una clave que ya existe**
- **Insertar un empleado que no existe.**
- **Cualquier otra excepción que consideres conveniente.**

```
CREATE TABLE T_LIQUIDACION (
   apellido VARCHAR2(40),
   departamento VARCHAR2(40),
   oficio VARCHAR2(50),
   salario NUMBER(7,2),
   trienios NUMBER(7,2),
   comp_responsabilidad NUMBER(7,2),
   comision NUMBER(7,2),
   total NUMBER(7,2),
   PRIMARY KEY (apellido)
);

CREATE OR REPLACE PROCEDURE guardar_liquidacion(p_emp_no INT) IS
   v_emp emple%ROWTYPE;
   v_dep_nombre VARCHAR2(40);
   v_trienios NUMBER;
   v_comp_responsabilidad NUMBER;
   v_total NUMBER;
BEGIN
   SELECT * INTO v_emp FROM emple WHERE emp_no = p_emp_no;
   SELECT dnombre INTO v_dep_nombre FROM depart WHERE dept_no = v_emp.dept_no;

   v_trienios := FLOOR(MONTHS_BETWEEN(SYSDATE, v_emp.fecha_alt) / 36) * 30;
   SELECT COUNT(*) * 60 INTO v_comp_responsabilidad FROM emple WHERE dir =
v_emp.emp_no;
   v_total := v_emp.salario + v_trienios + v_comp_responsabilidad + NVL(v_emp.comision, 0);

   INSERT INTO T_LIQUIDACION VALUES (v_emp.apellido, v_dep_nombre, v_emp.oficio,
v_emp.salario, v_trienios, v_comp_responsabilidad, NVL(v_emp.comision, 0), v_total);
   COMMIT;
EXCEPTION
   WHEN DUP_VAL_ON_INDEX THEN
      DBMS_OUTPUT.PUT_LINE('Error: El registro ya existe.');
END;
/
```

13. Si quisieras controlar la excepción que sale al llamar al procedimiento anterior pasando como parámetro un tipo de dato erróneo, por ejemplo, una cadena de texto en lugar de un entero, ¿cómo lo harías?

```
SET SERVEROUTPUT ON
BEGIN
        liquidar('a');
EXCEPTION
        WHEN VALUE_ERROR THEN
                DBMS_OUTPUT.PUT_LINE('VALOR ERRONEO');
END;
/
```

EXCEPCIONES

1. Realiza una función BUSQ_NOM_DEP que, a partir del dept_no, devuelva el nombre del departamento. Utilizad las excepciones para detectar cuando no existe un departamento.

```
CREATE OR REPLACE FUNCTION BUSQ_NOM_DEP(p_dept_no IN DEPART.dept_no%TYPE)
RETURN DEPART.dnombre%TYPE IS
  v_nombre DEPART.dnombre%TYPE;
BEGIN
  -- Buscar el nombre del departamento
  SELECT dnombre INTO v_nombre FROM DEPART WHERE dept_no = p_dept_no;
  RETURN v_nombre;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN 'Departamento no encontrado.';
  WHEN OTHERS THEN
    RETURN 'Error: ' || SQLCODE ||SQLERRM;
END ;
/

BEGIN
  DBMS_OUTPUT.PUT_LINE(BUSQ_NOM_DEP(1));
END;
/
```

2. Realiza un procedimiento BUSQ_EMP para que a partir de un emp_no, busque y visualice todos los datos del empleado. Utilizar las excepciones para detectar cuando no existe un empleado.

```
CREATE OR REPLACE PROCEDURE BUSQ_EMP(p_emp_no IN EMPLE.emp_no%TYPE) IS
   v_empleado EMPLE%ROWTYPE;
BEGIN
   SELECT * INTO v_empleado FROM EMPLE WHERE emp_no = p_emp_no;
   DBMS_OUTPUT.PUT_LINE('Empleado: ' || v_empleado.apellido || ', Oficio: ' || v_empleado.oficio
||  ', Salario: ' || v_empleado.salario || ', Departamento: ' || v_empleado.dept_no);
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Error: Empleado no encontrado.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error inesperado: ' || SQLCODE || SQLERRM);
END ;
/

BEGIN
   BUSQ_EMP(1);
END;
/
```

3. Crea una función CALC_SUELDO que a partir de un número de empleado calcule su sueldo, siendo éste su salario + nómina. Recordad que si la nómina es nul1 será necesario convertirla a número con la función NVL. Debe de controlarse la excepción en caso de que no exista el empleado.

```
CREATE OR REPLACE FUNCTION CALC_SUELDO(p_emp_no IN EMPLE.emp_no%TYPE)
RETURN NUMBER IS
   v_sueldo NUMBER;
BEGIN
   SELECT salario + NVL(comision, 0) INTO v_sueldo FROM EMPLE WHERE emp_no =
p_emp_no;
   RETURN v_sueldo;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
         DBMS_OUTPUT.PUT_LINE( 'Empleado no encontrado.');
          RETURN -1;
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error inesperado: ' || SQLCODE || SQLERRM);
          RETURN -1;
END;
/
```

4. Crea un procedimiento que, a partir de una cantidad en euros pasada por parámetro, actualice la comisión de todos los empleados de un departamento. Dicha cantidad podrá ser positiva o negativa. Además, se deberá considerar lo siguiente:

- **Si el departamento en el que se realiza la actualización no existe, se deberá de mostrar un mensaje de error.**
- **Del mismo modo, se controlará la excepción que pueda surgir si la comisión llega a ser negativa.**
- **Se deberá de mostrar los datos datos del empleado con la comisión actualizada, y a su vez, el sueldo total que posee.**

```
CREATE OR REPLACE PROCEDURE ACTUALIZAR_COMISION(p_dept_no IN
EMPLE.dept_no%TYPE, p_cantidad IN NUMBER) IS
BEGIN
   UPDATE EMPLE SET comision = NVL(comision, 0) + p_cantidad WHERE dept_no = p_dept_no;
   IF SQL%ROWCOUNT = 0 THEN
      RAISE_APPLICATION_ERROR(-20003, 'Departamento no encontrado.');
   END IF;
   DBMS_OUTPUT.PUT_LINE('Comisión actualizada.');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/


CREATE OR REPLACE PROCEDURE ACTUALIZAR_COMISION(
   p_dept_no IN EMPLE.dept_no%TYPE,
   p_cantidad IN NUMBER
) IS
   v_existe NUMBER;
   e_comision_negativa EXCEPTION;
BEGIN
   -- Verificar si el departamento existe
   SELECT COUNT(*) INTO v_existe FROM EMPLE WHERE dept_no = p_dept_no;

   IF v_existe = 0 THEN
      RAISE_APPLICATION_ERROR(-20003, 'Error: Departamento no encontrado.');
   END IF;

   -- Actualizar la comisión de los empleados
   UPDATE EMPLE
   SET comision = NVL(comision, 0) + p_cantidad
   WHERE dept_no = p_dept_no;

   -- Verificar si alguna comisión es negativa
```

```
    IF EXISTS (SELECT 1 FROM EMPLE WHERE dept_no = p_dept_no AND comision < 0) THEN
        ROLLBACK;
        RAISE e_comision_negativa;
    END IF;

    -- Mostrar datos actualizados
    FOR rec IN (
        SELECT emp_no, nombre, salario, comision, (salario + NVL(comision, 0)) AS sueldo_total
        FROM EMPLE
        WHERE dept_no = p_dept_no
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Empleado: ' || rec.emp_no || ' - ' || rec.nombre);
        DBMS_OUTPUT.PUT_LINE('Salario: ' || rec.salario || ' - Comisión: ' || NVL(rec.comision, 0));
        DBMS_OUTPUT.PUT_LINE('Sueldo Total: ' || rec.sueldo_total);
        DBMS_OUTPUT.PUT_LINE('----------------------------');
    END LOOP;

    COMMIT;
EXCEPTION
    WHEN e_comision_negativa THEN
        DBMS_OUTPUT.PUT_LINE('Error: La comisión no puede ser negativa.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error inesperado: ' || SQLERRM);
        ROLLBACK;
END;
/
```

5. Realiza un script que permita insertar un empleado en un departamento, una vez insertado debe de mostrar un mensaje indicándolo. A su vez, se deberá de controlar los posibles errores que puedan surgir, como por ejemplo:

- **El número de empleado ya existe.**
- **Se insertan valores nulos en campos "no nulos", como por ejemplo apellido.**
- **Se inserta una clave ajena errónea, como puede ser departamento o un jefe que no existe.**
- **Se indica un oficio que no es posible en el departamento, como por ejemplo "soldador".**
- **Se introduce un tipo de dato erróneo, como puede ser un varchar en el salario.**

```
CREATE OR REPLACE PROCEDURE INSERTAR_EMPLEADO(
   p_emp_no IN EMPLE.emp_no%TYPE,
   p_apellido IN EMPLE.apellido%TYPE,
   p_oficio IN EMPLE.oficio%TYPE,
   p_dir IN EMPLE.dir%TYPE,
   p_fecha_alt IN EMPLE.fecha_alt%TYPE,
   p_salario IN EMPLE.salario%TYPE,
   p_comision IN EMPLE.comision%TYPE,
   p_dept_no IN EMPLE.dept_no%TYPE
) IS
BEGIN
   INSERT INTO EMPLE VALUES (p_emp_no, p_apellido, p_oficio, p_dir, p_fecha_alt,
p_salario, p_comision, p_dept_no);
   DBMS_OUTPUT.PUT_LINE('Empleado insertado correctamente.');
EXCEPTION
   WHEN DUP_VAL_ON_INDEX THEN
      DBMS_OUTPUT.PUT_LINE('Error: El número de empleado ya existe.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

6. Crea un procedimiento que elimine un empleado, se debe de mostrar un mensaje indicando: Si se ha borrado correctamente. Si dicho empleado no existe en la tabla.

```
CREATE OR REPLACE PROCEDURE ELIMINAR_EMPLEADO(p_emp_no IN
EMPLE.emp_no%TYPE) IS
BEGIN
   DELETE FROM EMPLE WHERE emp_no = p_emp_no;
   IF SQL%ROWCOUNT = 0 THEN
      DBMS_OUTPUT.PUT_LINE('Empleado no encontrado.');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Empleado eliminado correctamente.');
   END IF;
EXCEPTION
WHEN EMPLE_NOT_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Empleado no encontrado’);
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error al eliminar el empleado: ' || SQLERRM);
END;/
```

7. Crea un procedimiento llamado cambiarJefe que cambie el director o jefe de un empleado en la tabla emple de la base de datos Empresa.
Este procedimiento recibirá el apellido del empleado cuyo jefe se desea modificar y el apellido de su nuevo empleado jefe.
En este procedimiento se pueden dar dos situaciones de error o excepciones:

- **Que no exista el empleado cuyo jefe se desea modificar.**
- **Que no exista el nuevo empleado jefe que se le quiere asignar.**

Se desea que se distinga cuando da una excepción u otra.

```
CREATE OR REPLACE PROCEDURE CAMBIAR_JEFE(
   p_emp_apellido IN EMPLE.apellido%TYPE,
   p_nuevo_jefe IN EMPLE.apellido%TYPE
) IS
   v_emp_no EMPLE.emp_no%TYPE;
   v_nuevo_jefe_no EMPLE.emp_no%TYPE;
BEGIN
   SELECT emp_no INTO v_emp_no FROM EMPLE WHERE apellido = p_emp_apellido;
   SELECT emp_no INTO v_nuevo_jefe_no FROM EMPLE WHERE apellido = p_nuevo_jefe;

   UPDATE EMPLE SET dir = v_nuevo_jefe_no WHERE emp_no = v_emp_no;
   DBMS_OUTPUT.PUT_LINE('Jefe cambiado correctamente.');
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Error: Empleado o jefe no encontrado.');
--hay que diferenciar jefe o empleado.
   WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;/

CREATE OR REPLACE PROCEDURE CAMBIAR_JEFE(
  p_emp_apellido IN EMPLE.apellido%TYPE,
  p_nuevo_jefe IN EMPLE.apellido%TYPE
) IS
  v_emp_no EMPLE.emp_no%TYPE;
  v_nuevo_jefe_no EMPLE.emp_no%TYPE;
  v_count NUMBER;
BEGIN
  -- Verificar si el empleado existe
  SELECT COUNT(*) INTO v_count FROM EMPLE WHERE apellido = p_emp_apellido;
  IF v_count = 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Error: No existe el empleado ' || p_emp_apellido);
  END IF;

  -- Obtener el emp_no del empleado
  SELECT emp_no INTO v_emp_no FROM EMPLE WHERE apellido = p_emp_apellido;

  -- Verificar si el nuevo jefe existe
  SELECT COUNT(*) INTO v_count FROM EMPLE WHERE apellido = p_nuevo_jefe;
  IF v_count = 0 THEN
    RAISE_APPLICATION_ERROR(-20002, 'Error: No existe el nuevo jefe ' || p_nuevo_jefe);
  END IF;

  -- Obtener el emp_no del nuevo jefe
  SELECT emp_no INTO v_nuevo_jefe_no FROM EMPLE WHERE apellido = p_nuevo_jefe;

  -- Actualizar el jefe del empleado
  UPDATE EMPLE SET dir = v_nuevo_jefe_no WHERE emp_no = v_emp_no;

  -- Confirmación
  DBMS_OUTPUT.PUT_LINE('Jefe cambiado correctamente.');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

TRIGGERS / DISPARADORES

1. Crear un disparador asociado a la tabla EMPLE para que únicamente permita insertar empleados si es lunes, tomando como referencia la fecha del sistema.

```
CREATE OR REPLACE TRIGGER trg_insertar_emple_lunes
BEFORE INSERT ON emple
FOR EACH ROW
BEGIN
   IF TO_CHAR(SYSDATE, 'D') != 2 THEN -- =2 significa lunes, y D es para el día
      RAISE_APPLICATION_ERROR(-20100, 'Solo se pueden insertar empleados los lunes.');
   END IF;
END;
/
```

2. Diseñar una tabla llamada AUDITAR_DEPART con los siguientes campos: FECHA, CAMPO, OPERACIÓN, VALOR_VIEJO, VALOR_NUEVO

```
CREATE TABLE auditar_depart (
   fecha       DATE,
   campo       VARCHAR2(40),
   operacion   VARCHAR2(20),
   valor_viejo  VARCHAR2(100),
   valor_nuevo  VARCHAR2(100)
);
/
```

3. Diseñar un trigger que permita auditar cualquier operación de inserción en la tabla DEPART, siempre y cuando ésta se lleve a buen fin.

En este caso se deberá de insertar en la tabla AUDITAR_DEPART:

- **Fecha.**
- **El nombre de campo, en este caso el texto "DEPT_NO".**
- **Operación: el texto "INSERTADO".**
- **Valor_viejo: NULL, ya que es un departamento nuevo.**
- **Valor nuevo: deberá de aparecer el valor del DEPT_NO que se ha insertado.**

Realizad la inserción de un nuevo departamento en la tabla depart y comprobad si se ha ejecutado el trigger.

```
CREATE OR REPLACE TRIGGER trg_auditar_insert_depart
AFTER INSERT ON depart
FOR EACH ROW
BEGIN
   INSERT INTO auditar_depart (fecha, campo, operacion, valor_viejo, valor_nuevo)
   VALUES (SYSDATE, 'DEPT_NO', 'INSERTADO', NULL, TO_CHAR(:NEW.dept_no));
END;/
```

4. Modifica el trigger anterior, para que además de registrar la inserción de un departamento en la tabla AUDITAR_DEPART, se registre la eliminación.

El registro en este caso será igual que en la inserción, exceptuando el apartado operación que se habrá de poner el texto "BORRADO".

```
DROP TRIGGER trg_auditar_insert_depart; --borrar la anterior

CREATE OR REPLACE TRIGGER trg_auditar_insert_delete_depart --crear nuevo trigger
AFTER INSERT OR DELETE ON depart
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO auditar_depart (fecha, campo, operacion, valor_viejo, valor_nuevo)
    VALUES (SYSDATE, 'DEPT_NO', 'INSERTADO', NULL, TO_CHAR(:NEW.dept_no));
  ELSIF DELETING THEN
    INSERT INTO auditar_depart (fecha, campo, operacion, valor_viejo, valor_nuevo)
    VALUES (SYSDATE, 'DEPT_NO', 'BORRADO', TO_CHAR(:OLD.dept_no), NULL);
  END IF;
END;
/
```

5.Vuelve a modificar el trigger anterior para que permita auditar cualquier modificación en la tabla DEPART, insertando en la tabla AUDITAR_DEPART los siguientes datos:

- **FECHA: Fecha del sistema**
- **CAMPO: nombre del campo del registro sobre el que se está actuando • OPERACIÓN: el texto "MODIFICACIÓN"**
- **VALOR_VIEJO: valor antiguo**
- **•VALOR_NUEVO: valor asignado al nuevo campo**

Se debe de comprobar que realiza todos los casos, incluido el modificar la clave primaria de un departamento que tiene empleados asociados, los cuales, deberán actualizar dicho campo en la tabla emple.

```
DROP TRIGGER trg_auditar_insert_delete_depart;

CREATE OR REPLACE TRIGGER trg_auditar_todo_depart
AFTER INSERT OR DELETE OR UPDATE ON depart
FOR EACH ROW
DECLARE
  v_campo VARCHAR2(40);
BEGIN
  IF INSERTING THEN
    INSERT INTO auditar_depart (fecha, campo, operacion, valor_viejo, valor_nuevo)
    VALUES (SYSDATE, 'DEPT_NO', 'INSERTADO', NULL, TO_CHAR(:NEW.dept_no));
  ELSIF DELETING THEN
    INSERT INTO auditar_depart (fecha, campo, operacion, valor_viejo, valor_nuevo)
    VALUES (SYSDATE, 'DEPT_NO', 'BORRADO', TO_CHAR(:OLD.dept_no), NULL);
  ELSIF UPDATING THEN
    IF :OLD.dept_no != :NEW.dept_no THEN
      -- Auditoría del cambio
      INSERT INTO auditar_depart (fecha, campo, operacion, valor_viejo, valor_nuevo)
      VALUES (SYSDATE, 'DEPT_NO', 'MODIFICACIÓN', TO_CHAR(:OLD.dept_no),
TO_CHAR(:NEW.dept_no));

      -- Actualizar la clave foránea en EMPLE
      UPDATE emple
      SET dept_no = :NEW.dept_no
      WHERE dept_no = :OLD.dept_no;
    END IF;

    IF :OLD.dnombre != :NEW.dnombre THEN
      INSERT INTO auditar_depart (fecha, campo, operacion, valor_viejo, valor_nuevo)
      VALUES (SYSDATE, 'DNOMBRE', 'MODIFICACIÓN', :OLD.dnombre, :NEW.dnombre);
    END IF;
```

```
      IF :OLD.loc != :NEW.loc THEN
         INSERT INTO auditar_depart (fecha, campo, operacion, valor_viejo, valor_nuevo)
         VALUES (SYSDATE, 'LOC', 'MODIFICACIÓN', :OLD.loc, :NEW.loc);
      END IF;
   END IF;
END;
/
```

Más corto:

```
CREATE OR REPLACE TRIGGER trg_auditar_depart
AFTER UPDATE ON DEPART
FOR EACH ROW
DECLARE
   v_fecha     DATE := SYSDATE;
   v_operacion VARCHAR2(20) := 'MODIFICACIÓN';
BEGIN
   -- Comparar y auditar cada campo relevante
   IF :OLD.nombre != :NEW.nombre THEN
      INSERT INTO AUDITAR_DEPART VALUES (v_fecha, 'NOMBRE', v_operacion, :OLD.nombre,
:NEW.nombre);
   END IF;

   IF :OLD.ubicacion != :NEW.ubicacion THEN
      INSERT INTO AUDITAR_DEPART VALUES (v_fecha, 'UBICACION', v_operacion,
:OLD.ubicacion, :NEW.ubicacion);
   END IF;

   IF :OLD.dept_no != :NEW.dept_no THEN
      INSERT INTO AUDITAR_DEPART VALUES (v_fecha, 'DEPT_NO', v_operacion,
TO_CHAR(:OLD.dept_no), TO_CHAR(:NEW.dept_no));

      -- Actualizar clave en empleados relacionados
      UPDATE EMPLE
      SET dept_no = :NEW.dept_no
      WHERE dept_no = :OLD.dept_no;
   END IF;
END;
```

Profe:

```
CREATE OR REPLACE TRIGGER trg_auditar_todo_depart
AFTER INSERT OR DELETE OR UPDATE ON depart
FOR EACH ROW
BEGIN
   IF INSERTING THEN
      INSERT INTO auditar_depart VALUES (SYSDATE, 'DEPT_NO', 'INSERTADO',
NULL,:NEW.dept_no);
   ELSIF DELETING THEN
      INSERT INTO auditar_depart VALUES (SYSDATE, 'DEPT_NO', 'BORRADO',:OLD.dept_no,
NULL);
   ELSIF UPDATING (‘dept_no’)THEN
         UPDATE emple
         SET dept_no = :NEW.dept_no
         WHERE dept_no = :OLD.dept_no;
     INSERT INTO auditar_depart VALUES (SYSDATE, 'DNOMBRE', 'MODIFICACIÓN',
:OLD.dnombre, :NEW.dnombre);

ELSIF UPDATING (‘DNOMBRE’)THEN
    INSERT INTO auditar_depart VALUES (SYSDATE, ‘DNOMBRE’, 'MODIFICADO',
:OLD.dnombre, :NEW.dnombre);
ELSIF UPDATING (LOC)THEN
    INSERT INTO auditar_depart VALUES (SYSDATE, LOC, 'MODIFICADO', :OLD.loc, :NEW.loc);
     END IF;
END;
/
```